10-1-1999

Expectation Formation in Multi-Agent Design Systems

Dan L. Grecu
Worcester Polytechnic Institute, dgrecu@cs.wpi.edu

David Brown
Worcester Polytechnic Institute, dcb@wpi.edu

EXPECTATION FORMATION IN MULTI-AGENT DESIGN SYSTEMS

DAN L. GRECU, DAVID C. BROWN
AI in Design Group
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609, U.S.A.
E-mail: dgrecu@cs.wpi.edu, dcb@cs.wpi.edu

1.0 Introduction

It has become obvious in recent years that design systems need, or at least can benefit from, some type of learning (Duffy 1997). So far the effort to bring learning into design systems has addressed this need in a very specific way. Developers[1] have singled out *a priori* a particular learning target, and have shaped the design system to acquire data for the learning task. Thus, a developer decided beforehand that the design system would learn how to classify (Reich and Fenves 1991); how to rank designs (Murdoch and Ball 1994); that it would develop relations between design concepts (Maher and Li 1994); or that it would acquire descriptions that lead to structurally optimal designs. Depending on the predefined learning task, design systems were 'crafted' to collect the appropriate information to achieve their learning goal.

The learning approach described above responds to two types of demands formulated by developers and designers:

1. It may be the case that the *designer* needs design information or designing information that is otherwise not available, and therefore requests that such information be learned by the design system. Situations of this type are often encountered when the designer needs to solve optimization problems for which data is hard to acquire and/or process (Figure 1a).
2. As design systems become more complex, the *developers* become those that are in need of information. If a developer finds it difficult to provide a certain kind of knowledge when crafting the design system, he/she has the option of building a learning component into the design system that will acquire the missing knowledge (Figure 1b). The learned knowledge is typically designing knowledge (problem-solving and other design process knowledge) that the design system will use in generating future designs.

---

1. In this paper we refer to the experts in charge of creating design systems as *developers*, and to the experts that create design artifacts in a particular design domain as *designers.*

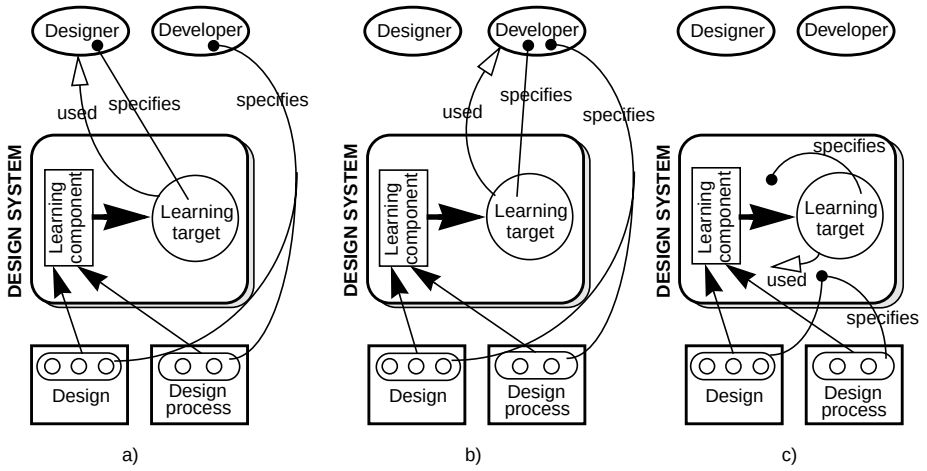

Figure 1. Control and use of learning in a design system

In both cases learning is procedural, i.e., it is implemented through procedures that always use the same sources of information for learning, and are always geared towards the same type of learning target(s).

What if the designer requires other types of information to be learned? How can a developer, who realizes that he/she needs to provide the design system with new kinds of design information, redirect the learning component towards new learning targets? The answer lies in *re-engineering* the design system's learning procedure so that it can capture new types of training data, and use it to classify or predict new types of design or design process values.

Another important observation is that in both cases the learning supports the human. In the first case the learning component supplies the designer with technical design data, whereas in the second case it serves as an acquisition tool for information items indicated by the developer.

In this paper we investigate a 'learning in design' model that differs substantially from previous models. In this work, learning has a strong declarative dimension, as the sources for the learning information and the learning targets are *not* built into the learning mechanism. They are specified in a declarative manner, and are the result of other reasoning processes in the design system.

This flexibility in instantiating a learning mechanism in various contexts provides the challenge of how to leave the control of the learning in the 'hands' of the design system (Figure 1c). We are investigating how a design system can use the learning towards targets that it selects *itself*. To demonstrate the autonomy of the learning, independent from the designer and developer, we will also show that

the proposed learning model enables a design system *to search and find information sources* that will support the learning processes it has initiated.

To provide a strong motivation for this type of learning we will place our discussion in the context of multi-agent design systems (MADS), where the criteria for defining the exact dimensions of learning processes from the beginning are particularly hard to define.

The rest of the paper proceeds by presenting the need for learning, and the nature of design agents and their decision-making. We then discuss the use of expectations, as well as when and how they can be learned. Finally, a system called LEAD is presented and experiments with it are discussed.

2.0 Distributed Design

2.1 FROM EVALUATION NEEDS TO LEARNING

As the complexity of design problems that fall into the realm of AI steadily increases, more and more design systems are developed as distributed environments, whether as collaborative design systems (Malone 1998), or as multi-agent design systems (Lander 1998). Collaborative design systems operate at the high end of design complexity, and attempt to integrate the work of human experts that contribute to the development of a product. Multi-agent systems focus on small scale design problems, however with little or no human involvement during the design process.

Both approaches involve considerable integration problems. The ‘designers’, be they humans or agents, proceed with their individual problem solving task, only to realize at some point that their partial solutions are inconsistent, or do not achieve the desired design requirements. Design expertise is not additive, in the sense that simply the presence of *all* the domain design knowledge needed to solve a design problem does not create a skilled ‘designer’, whether this designer is a design team or a multi-agent design system.

Good designing relies, amongst other things, on the ability to relate design decisions with the rest of the design, both temporally and spatially. Isolated design takes an immediate perspective, where a decision can be adopted if its preconditions are met at *the current time point* by the information available in *the limited design context* of a given designer. In contrast, skilled designers often take a decision before all the underlying information becomes available, by relying on good (perhaps heuristic) evaluation mechanisms to compensate for the missing information. They are able to look at the impact of the decision further “down the road”, and to rule out decision options that may fail. They are also able to consider the ramifications of the preconditions of the decision, and look at elements from the global design environment that may cause the decision to fail.

Clearly, all the skills outlined above rely on some kind of evaluation. To take decisions early, before all the components of the decision are confirmed, requires consideration of these missing components. To weigh the potential problems or

benefits of a decision, one needs to evaluate the impact of the decision by predicting the values with which the decision may come in conflict, or by predicting the goals that it may 'serve'. Experts resort to the integration of preliminary evaluations made with partial data into the decision-making heuristic. This enhances a decision's sensitivity to the larger design context,

The ability to evaluate can be learned. In this learning task the learning target is the object of the evaluation. Consider, for example, the need to evaluate the area of a chair seat during chair design before the parameter values that define the area have been decided. The designer has to identify the design or design process elements available at that particular point on which to base the evaluation.

Will the chair seat be supported by one central foot, by three, or by four legs? Will the seat be curved or flat? Will the seat be made of wood, plastic or metal? Does the maximal cost of the chair have an impact on the surface of the seat? All these factors, once pruned to a relevant set, will represent potential indicators that might be used to predict the seat area. An experienced designer will know how to quickly acquire representative values for these indicators, allowing him/her to make good predictions for the chair seat area early in the design process.

Little in a design *problem* indicates that there will be a need to evaluate the area of a chair seat. This need will become clear only after repeated design sessions. Therefore, the need for a learning component in the design system, capable of handling this type of evaluation, will arise from the *designing process,* i.e., only after the developers have completed the design system.

How does this analysis relate to the discussion of the learning approaches in the first section? First of all, as the next section will point out in more detail, learning clearly becomes an integration tool in a distributed design system, and this remains true whether the system is a collaborative design environment or a multi-agent design system. Distribution eliminates the presence of a complete global image of the design, and it also abolishes the single designing perspective. Evaluation and learning become tools to regain some of the advantages lost to the fragmentation of designing.

Since we are interested in design systems that are completely autonomous in their design task, we will restrict our further discussion to multi-agent systems. Under this assumption it is clear that the type of learning we are looking at is intended to support the design system itself. The learning component neither provides additional design information to an external designer, nor does it directly respond to developers' *pre-specified* knowledge insertion goal through learning. The essential elements of the learning process are to be determined by the agents themselves: *what* to learn about, *where* to look for supporting information, and *when* to initiate a learning process in the first place.

2.2 THE DESIGN AGENT WORLD

Consider a multi-agent system for basic chair design. Agents are in charge of various aspects of designing, such as seat design, backrest design, frame design,

and assessment based on ergonomic and cost criteria. Design agents have *specialized* knowledge about the problem domain in which they operate. Based on the tasks they execute, *agent functionalities* include *decision-making* about the design components (e.g., seat, backrest and frame), and *critiquing* of design aspects (e.g., ergonomics and cost).

Within its 'society' an agent may know about the roles or specializations of the agents with which it interacts, about when to act, how to communicate, and how to solve conflicts with other agents. However, it is not realistic to expect an agent to be able to anticipate or to compute the behavior of all the other agents in the system (Cherniak 1986; Russell and Wefald 1991).

We would like to have a system where design agents base their decisions on *all* the knowledge that is available in the design system, and where they know the possible consequences of *every* potential decision. The utilities associated with these consequences would drive the decision selection, and would allow agents to precisely respond to design goals (Figure 2).

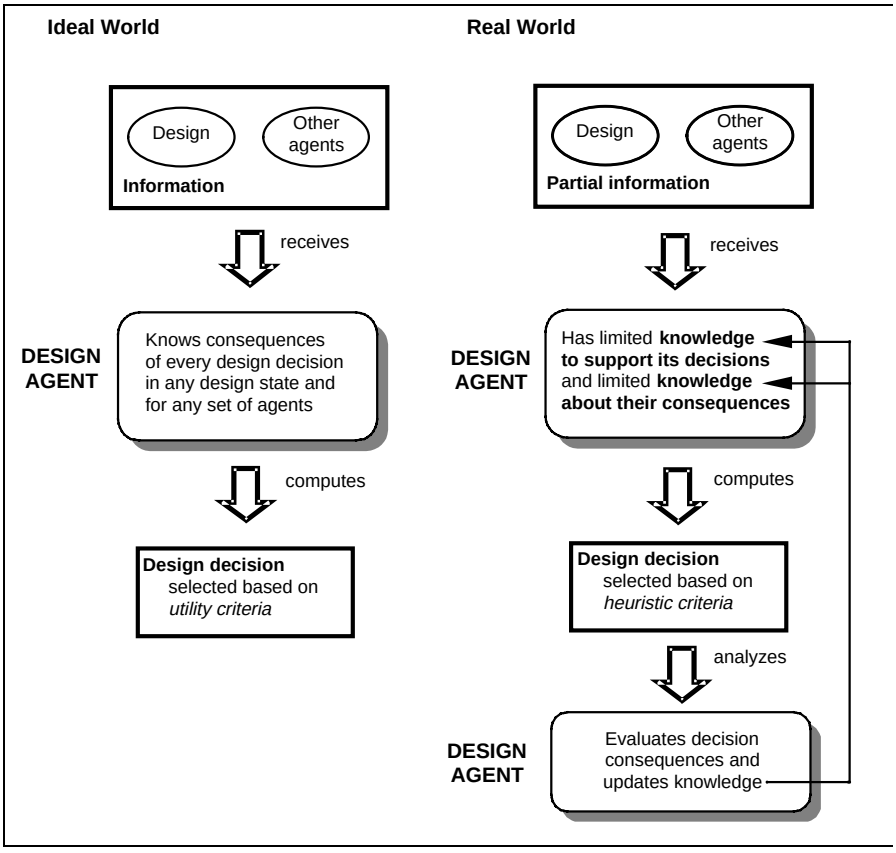

Figure 2. Agent decision-making in design

In reality agents have only limited information about how other agents operate, about their knowledge, and their internal reasoning strategies. Furthermore, agents typically see only the part of the design covered by their domain competence. As a result, agents base their decisions on the knowledge they *have*, and not on the knowledge that is available in the system. After a decision has been taken an agent may *sometimes* know *some* of the consequences of a decision it has made, but it cannot know or compute all the consequences of its decisions. Decision making and decision analysis require some means to compensate for information that is not available at that point or cannot be deduced.

The difference between the "ideal" and the "real" setting in a multi-agent design system calls for the support of learning to acquire knowledge that can be helpful in the design process. This knowledge is inherently heuristic, since it results from design experience. Human designers perceive the need to learn, delimit the learning setting, and accumulate observations with considerable ease. Neither of these skills is a trivial task for a design agent. We now describe a model that implements precisely these features into a design agent.

3.0 Expectations in Design

3.1 USING EXPECTATIONS IN DESIGN

Expectations are a form of empirically derived knowledge that compensate for the absence of deductively derived knowledge. Expectations express the belief that an event will happen, and describe the circumstances or conditions under which that event will happen. They are typically created because limited resources, such as time or information, prevent the holder of the expectation from establishing a proven causal relationship between the set of conditions and the ensuing situation.

In a multi-agent design system, expectations represent the knowledge of agents that events will occur in a pre-defined way: for example, that design parameters will be within specific ranges, that responses from other agents will arrive within a given amount of time, or that decisions will lead to given outcomes. Figure 3 shows an example of an expectation, expressed as a rule. The conditions for the cost expectation include conditions related to the design and to the design agents.

Expectations are precisely the form of knowledge one would like to have when a decision needs to be taken, and some of the pre-conditions for that decision have not yet been confirmed. Alternatively, one might use expectations to determine the consequences of a decision. For example, an agent might use an expectation, such as the one described in figure 3, when deciding the frame material for a chair to make sure that a cost constraint will not be violated.

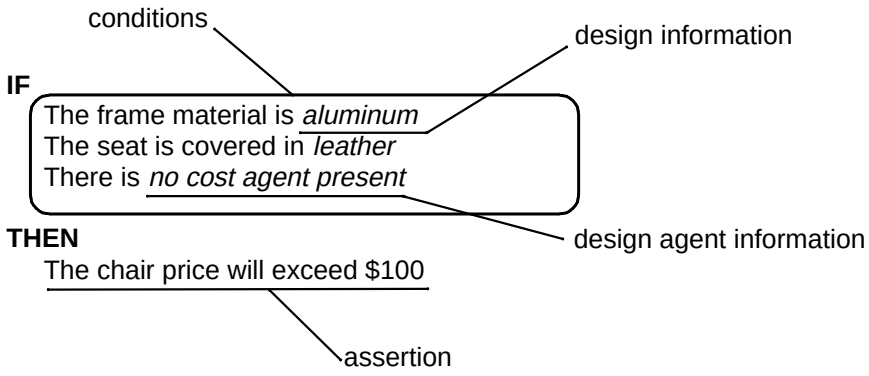

Figure 3. Design expectation example

Expectations are of particular benefit in a multi-agent system where agents would be otherwise isolated in their own domain niches. There are two aspects to expectations that makes them particularly appealing. First, from the point of view of their contents, they tend to combine information from outside of an agent's own realm. Expectations are typically formed in an area where an agent does not have the ability to reason in detail. Second, expectations are always generated in response to an information need. This guarantees that a learning process based on expectations can be confined within semantically meaningful limits, and that it will not attempt to acquire information that is questionable from the point of view of its usability.

3.2 LEARNING EXPECTATIONS

3.2.1 Human learning

The task of acquiring expectations is not as intuitive as are the possibilities for their use. From the very beginning we should make a clear distinction between *learning expectations* and *learning from expectation violations*. The latter is a topic which has been investigated in some well-known models, such as the Rescorla-Wagner psychological theory (1972) that states that "organisms only learn when events violate expectations," and in Roger Schank's model of dynamic memory (1984) that contends that expectation failures prompt humans to memorize new information, and describes how expectations can be revised through explanation processes.

There is considerably less work that attempts to explain how expectations are acquired in the first place. In psychological research, the speed with which expec-

tations are generated and applied by humans has led to the conclusion that the process of expectation formation and use is not highly deliberative. Research in this direction has proven very difficult since subjects "while engaging in interaction, typically are not aware of how expectation states are formed, what states are formed, or how these states are transformed into behavior" (Berger et al. 1985).

Expectations have been strongly tied to the ability to work in teams. Wittenbaum et al. (1996) describe how working groups develop a tacit coordination through "the synchronization of the members' actions based on unspoken assumptions about what others in the group are likely to do" (p. 129). Their research shows that expectations refer to a task domain, including the steps that are part of the task, the goals that are pursued, and the quality criteria. They also refer to the participants in the task, the decisions they take, and the circumstances under which they act or react. One of the important conclusions they draw is that expectations are developed based on carefully selected cues, and are not simply associations between any factors that might correlate with the target of the expectation.

Expectation learning requires the identification of the *conditions* that predict values for the object on which the expectation focuses, i.e., the *target* of the expectation. Statements that assign values or ranges to the target are called a*ssertions*. Expectation learning amounts to a 'causal reasoning' process – a search for the conditions that might influence the assertion. Recent research in understanding the mechanisms that underlie causal reasoning has identified two major stages within this process: the use of causal mechanisms to delimit a set of *candidate conditions*, and the use of covariational principles to extract from the candidate conditions the subset that is *relevant* for predicting the assertion (Koslowski 1996).

The causal mechanisms involved in the first stage of the expectation learning process play a fundamental role in focusing the learning process. A pure covariational process would be simply overwhelmed by the number of influence factors it would have to consider. It has been argued that people rely only on statistical associations to identify causes and explain events, and deviations from this behavior were regarded as cognitive biases (Tversky and Kahneman 1974). A significant body of evidence indicates that this is the case only when any other evidence or information is lacking. However, domain experts tend to go through a causal attribution stage in which they use domain specific knowledge to reason about possible causes for an event, prior to proceeding to a do correlation analysis between the variations in the conditions and the variation of the expectation assertion (Hilton 1990; Koslowski 1996; Shultz et al. 1986).

3.2.2 When do agents acquire expectations

We first have to decide *when* a design agent is going to initiate an expectation learning process. Our approach is to have design agents learn in response to a

repeated, specific need for information during designing. This need for information is defined in a domain-independent manner, and can fall into one of the following categories: a) Preparing the information for a design decision; b) Assessing the impact of a design decision.

a) Preparing the information for a design decision

Design agent decisions are based on information expressed as preconditions. If the design decision has to be taken before all the information needed to evaluate the preconditions is available, the design agent will use expectations to complete the information.

Examples of situations that require or could benefit from the use of expectations as substitutes for precondition information are:

1. The agent is required to provide a decision within a given time.
2. The decision-making information built into the agents is cyclic, and therefore one of the agents has to make a decision before all the needed preconditions are satisfied by information in the design environment.
3. Since the order of decisions influences the design and the design process, taking a decision earlier can benefit other agents that rely on the information resulting from the decision.

The range of situations is not limited to the ones presented above. Each of these situations in turn requires some comment.

- The first situation occurs if there is a partial ordering of the actions of design agents which is reflected in a design plan. In such cases, agents may be required to complete a task before another agent can proceed, and, if possible, will have to substitute for missing information.
- The second situation can be avoided only if a formal verification process can secure that there are no circular dependencies between the knowledge of the MADS agents. Such techniques are difficult to implement over an agent set. Circular dependencies typically occur because of design constraints that span several agents, and removing them amounts to a constraint problem-solving task across agents. If none of the agents uses an evaluation, it is possible that the design agents will make decisions which result in conflicts. Alternatively, MADS developers can compile out the circular dependencies by introducing estimates. However, this approach is subject to the types of limitations discussed in the first part of the paper.
- The use of expectations in this third case can significantly enhance the range of options that are available in terms of the configuring the overall design process. However, the expectation has to be a reliable substitute for the actual information lest its use actually represents an impediment to the design process.

a) Assessing the impact of a design decision

Design agents evaluate the consequences of their decisions by inferring ahead as to whether the decision value will satisfy constraints or support goals. In doing so, it is likely that some of the information required in the inference process is not yet available, and therefore the agent will attempt to substitute for it with an expectation.

Imagine the frame design agent, in our chair design problem, making a decision about the frame material. Before committing to the design decision the agent may verify whether the decision will satisfy cost constraints. Therefore it will need to know the conditions that influence the cost, and the specific correlations between the values for the determined conditions and the cost ranges. An expectation such as the one described in Figure 3 could be critical in validating the agent's decision before all the cost components are known. Alternatively, the frame design agent may take a decision which is perfectly valid at that point, that will be used by other agents, only to be later invalidated in a cost analysis process.

3.2.3 How agents acquire expectations

Once an agent decides to acquire an expectation it initiates a learning process. The agent knows the expectation target at this point, and will try: a) to identify features or conditions in the design environment that influence the expectation target, and b) to determine the condition *values* that predict given target values or ranges.

Accordingly, the design agent implements two-stage expectation learning (Figure 4). In the first stage, **causal attribution**, the agent uses causal mechanisms to select candidate conditions from the external world and from its own domain-specific knowledge that, in some combination, might affect the expectation target. Subsequently, these conditions are submitted to a **covariational analysis** to select the candidate *relevant* conditions, i.e., the subset impacting the target. In the process of covariational analysis the agent also determines the specific expectation values that predict target values.

a) Causal attribution

The determination of plausible causes for the modification of a design element is typically a domain dependent process. While some of the knowledge used in causal attribution is social knowledge about group processes, a considerable amount of knowledge is rooted in the specific design domain of the MADS.

Causal attribution is a decomposition and propagation process from the target to features in the design domain or in the designing process that are known at the point when the expectation learning process is initiated. The process of causal attribution is iterated on these features, until the propagation reaches features that are known at the moment when the expectation is needed. The knowledge that

supports the causal attribution process may ‘cross’ from features into the design domain to features in the designing process, or vice versa.

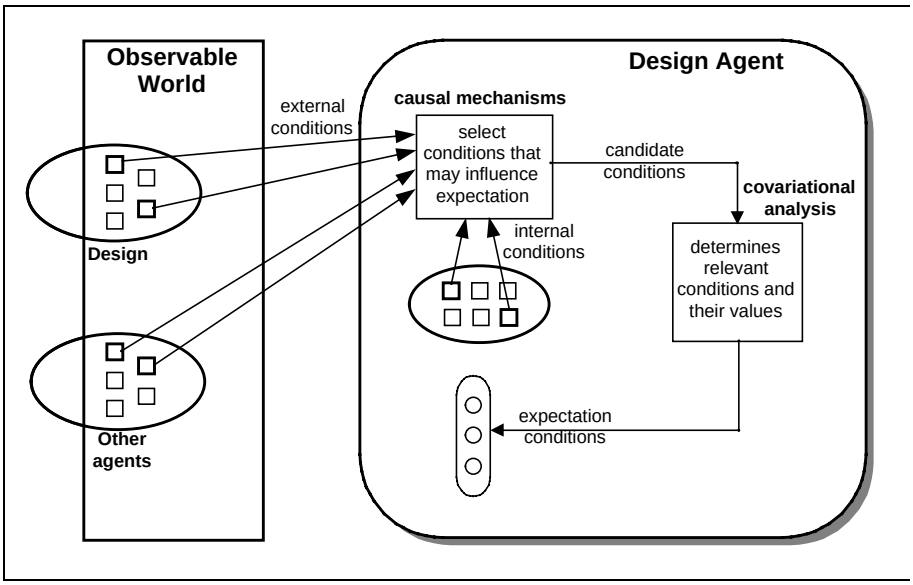

Figure 4. Expectation learning

Below we present examples of knowledge categories that we believe to be suitable for causal attribution in design problems. The use of specific types of knowledge depends on the knowledge representation, and on the reasoning processes implemented in the MADS.

1. *Structural design knowledge* supports the decomposition of causal attribution based on structural criteria. For example, if the target of the expectation is the weight of a chair, and the chair is composed of a frame, a backrest, and a seat, the causal attribution process will focus on these three design features as possible conditions for the expectation. The decision to proceed depends on whether the features are known at the point where the expectation is needed.

2. *Design features that share design constraints* with the expectation target provide candidates for the expectation condition since the modification of the constraint components is likely impact the target values.

3. *Representations of dependencies*, if available, provide a rapid method to elicit features influencing an expectation target. Belief net structures and influence diagrams explicitly introduce design features that are causally connected to the expectation target.

4. *Task decomposition* is useful in cases where an explicit task representation is available within the MADS, and when tasks result in the computation of design

features. Once a task is known to impact a design feature, the causal attribution process needs to identify the design elements that underlie the computation carried out by the task.

5. *Agent domains* provide another means to relate an expectation target with the agents' tasks or actions. *Agent functionalities*, when known to be relevant for deciding or agreeing on the value of the expectation target, can be used either to further look into their actions, or simply to relate the target value with their involvement in the design process (see example in Figure 3).

Overall, there is no universal set of knowledge entities to serve as a base for the causal attribution process. In this respect, the learning becomes dependent on the domain and functionality of the MADS, and needs to be supported during the development process. Since a some of the knowledge involved in causal attribution refers to the knowledge of the MADS itself, and is meta-knowledge, it often needs to be provided by the developer or there needs to be a capability to maintain and update this knowledge during the design process.

b) Covariational analysis

The covariational analysis is an inductive learning stage in which expectations are seen as concepts. The expectation conditions are the concept features, while the ranges for the expectation target values, such as the weight ranges of the chair in the previous example, represent the concept classes. The inductive learning algorithm learns a representation for the concept that will predict the class (range) of the expectation target from the values of the features identified as expectation conditions.

We should remember at this point that in the previous stage the design agent has identified a set of *candidate* conditions that it feels are relevant. This means that some of the conditions may not influence the expectation target at all. Other conditions may be redundant, and therefore can the pruned. Hence, the task of the covariational analysis is to determine a minimal subset of conditions that yield a sufficiently accurate prediction of the expectation target, *and* to learn the condition values that help make the classification of the target into ranges or classes (e.g., the price exceeds $100).

3.2.4 How agents validate expectations

Expectations are set up empirically, and therefore some validation process is required before using them. During the validation process an expectation is used to make predictions wherever the expectation assertion is needed. Given that the use of the expectation may actually alter the designing process, the validation is carried out in two phases.

In a first phase the expectation is used for predictions at the moment where it is needed, however, designing proceeds as if the expectation had not been available. The value that was predicted by the expectation is then compared with the final value resulting from the design process. If the expectation is violated, that is,

if the resulting value does not match the predicted assertion, the agent needs to review the expectation. We call this phase *contextual validation*, since the expectation is validated in a designing context similar to the one where the need for the expectation was identified.

In a second phase, the expectation is not only used to make a prediction, but it is actually used in the design process. Again, the expected value is compared with the final value resulting from the design process. We call this phase *semantic validation*, since it proves the validity of the expectation in a wider context, that may have been modified by the use of the expectation itself.

Figure 5 illustrates how an expectation is reviewed if it does not match the outcome of the design process. However, the process is generic and gets applied differently depending on the phase where the expectation is rejected.

If the expectation is rejected in the first phase, contextual validation, the agent will collect additional training data from both the cases where the expectation succeeds (positive training instances), and where it fails to predict the value of the target resulting from the design (negative training instances). The concept description of the expectation is updated with the new training data. This first phase is again followed by the semantic validation process.

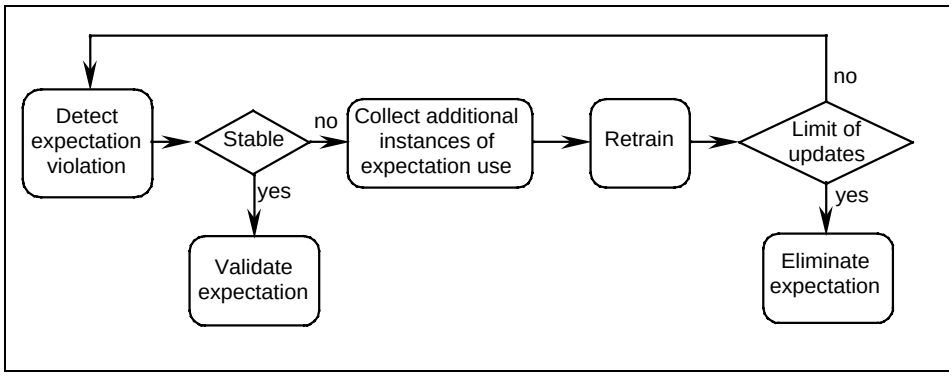

Figure 5. Expectation validation

If the expectation fails in the semantic validation phase, the agent proceeds to collect training cases from situations where the expectation is actually *used*. These training cases are added to the initial ones, where the expectation was learned without being used, and thus the expectation will cover both types of situations.

In both cases, the review process can be repeated for a pre-defined number of times. If the expectation does not reach a stable state, where no recent changes have been made, the agent will drop the expectation.

Several causes can prevent an expectation from being accepted. The causal mechanisms can lack sufficient coverage, preventing the inclusion of important discriminating conditions in the candidate set. Another possibility stems from the fact that several expectation learning processes can proceed simultaneously in several of the agents, thus changing their decisions and their behavior. If one of the changing elements associated with an agent is included among the conditions of an expectation that is developed by another agent, it is likely that this expectation will take a longer time to 'stabilize', or may lead to it being eliminated.

4.0 The LEAD System

LEAD is a system for Learning Expectations in Agent-based Design that was developed based on the framework described above and implemented in CLIPS (Giarratano and Riley 1998). In LEAD, agents act as design specialists and as group members. There are no agent hierarchies or relations between the agents that create rigid 'links' between them. However, the types of interactions between agents are predetermined, and they essentially represent the rules that create the group behavior. The interactions result dynamically, at run-time, and originate in the problem the system attempts to solve. The agents have complete autonomy in organizing their actions, with regard to the decisions they take as design specialists, or to their interactions with the rest of the group.

The agent model has evolved from the Single Function Agent (SiFA) paradigm (Dunskus et al. 1995), and includes specialized, knowledge-based design agents with precise functionality. Each agent has a predefined function in the design process. The agent types currently implemented in LEAD are:

- *Designers*: agents that are responsible for taking design decisions, such as selecting values for design parameters, or creating links between design components in a configuration process.

- *Critics*: agents that criticize design aspects, such as design parameter values, or weak properties of component configurations. Beyond revealing undesirable properties of the design, critics may point out constraints or quality requirements that are not met by the design aspect on which they focus.

- *Praisers*: agents that praise design aspects which rate particularly highly from a given point of view. Positive evaluations are important when designers have to decide which parts of the design need to be revised and which ones should preferably remain unchanged.

The agent function types are not necessarily limited to those described. The final application domain and the scale of the multi-agent system are the factors that ultimately decide the agent types to be included in the system.

All design agents have a restricted domain, the set of design elements that are the object of an agent's functionality. In parametric design problems, an agent's domain can be as narrow as a single design parameter. Several agents, of various

functionalities, can have overlapping domains. For example, the material for a component can be decided by a designer agent, and criticized by a cost critic.

Learning in LEAD is supported by two different components:

1) The *causal attribution component* identifies candidate conditions that may influence the expectation target. The primary body of knowledge underlying the causal attribution process is a model of the artefact being designed, including structural relationships, and function-structure relationships, describing the combinations of design parameters that help achieve specific functions, such as back support, stability, comfort etcetera in the example chair domain. The causal attribution knowledge also includes a description of the association between design parameters and agents. This allows LEAD to relate agent actions (e.g. decision, critique, request, conflict etc.) with the variation of a specific design parameter.

2) The *covariational analysis component* uses wrappers for relevant condition selection. Wrappers (Kohavi and John 1998; Liu and Setiono 1998) apply an induction algorithm to a training data set. The experiments are run by eliminating different sets of features from the training data instances. Specifically, wrappers eliminate conditions from the candidate condition set.

The wrapper method proposes a subset of features that are relevant for the identification of a given class. Features are considered relevant if their "values vary systemically with category membership" (Gennari, Langley, and Fisher 1989), in our case, with the ranges of the expectation target. For this purpose the wrapper maintains several subsets of candidate features. An accuracy testing component determines the performance of each subset, and eliminates or adds new subsets of features, by providing information to a feature selector.

Wrappers have the major advantage of being able to work with different learning algorithms, as long as the algorithms have the same interface. They have also been proven to be effective in pruning large initial sets of features (Kohavi and John 1998). Therefore, even if the agent does not have a strong set of causal mechanisms for setting up a new expectation and producing a small set of candidate conditions, the wrapper technique can partially compensate for this weakness. LEAD uses the MLC++ wrapper developed by SGI (1996).

5.0 A Design Problem

LEAD is currently applied to and tested using a parametric chair design problem. The reason for choosing this application domain is that besides the structural design issues, there are a considerable number of global constraints, which cannot be individually covered by any of the agents. These constraints stem primarily from the posture requirements for the human body, and from the use of the chair in conjunction with other furniture (table, desk, operator console etc.) for different functions: such as resting, writing, reading, manipulating controls, etc.

For example, Figure 6a indicates the relationship between optimum manual control areas and the angle of the backrest. Figure 6b shows the weight distribu-

tion for a particular angle of the backrest and seat position. This illustrates how structural computations can be strongly influenced by the type of activity performed by the person in the chair.

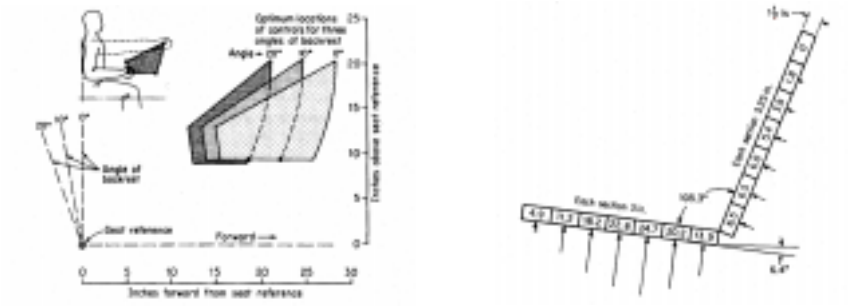

Figure 6. a) Optimum manual control areas in relation to angle of backrest;
b) Distribution of body weight on the universal test seat
for a given seat position and backrest angle (McCormick 1964)

Even for basic chairs, the requirements generated by the need to conform to the human body are complex, and vary depending on the height and age of the person. Designers try to respond to different categories of users and customers. Therefore they use different constraints and different computational methods to compute the chair's parameters for different users.

Ergonomic criteria relate these parameters to ensure basic standards for healthy seating. For example, one criterion restricts the amount of pressure that can be applied directly through the bones. This constraint includes the seat height, a chair frame parameter, as a person that has no support through the heels will have a higher percentage of his/her body weight supported by the seat.

It further depends on the angle between the backrest and the seat, also a frame parameter, as a closed angle raises the upper body to a vertical position, and increases pressure on the bones. The pressure can be reduced by the decision to use a lumbar support–a backrest parameter. The use of a padded seat, may further reduce the direct bone pressure.

This type of analysis illustrates some of the overarching constraints that characterize the chair design problem. Hence the agents' have an opportunity to make use of expectations to compensate for the part of the constraint that is 'invisible' to them and is handled by other agents.

6.0 Experiments with LEAD

For our design learning experiments we use a set of five agents: a seat design agent, a backrest design agent, a frame design agent, an ergonomic critic, and a

cost critic. The learning mechanisms are implemented only in the designer agents, since they are the only decision-making agents in LEAD.

To illustrate the system we will discuss an example of expectation learning extracted from LEAD. The example describes the acquisition of an expectation that is acquired in order to evaluate the consequences of a decision.

The ergonomic critic determines that the distance between the front side of the seat and the backrest is too large (Figure 7). This may have two causes: the depth of the seat (*s_depth*) is too large, or the distance between the back end of the seat and the seat reference is too large (*s_pos*).

The *s_depth* parameter is decided by the seat designer, the *s_pos* parameter is determined by the frame designer. It can be assumed that, since both agents have released these parameter values, they do not violate any of their individual design constraints.

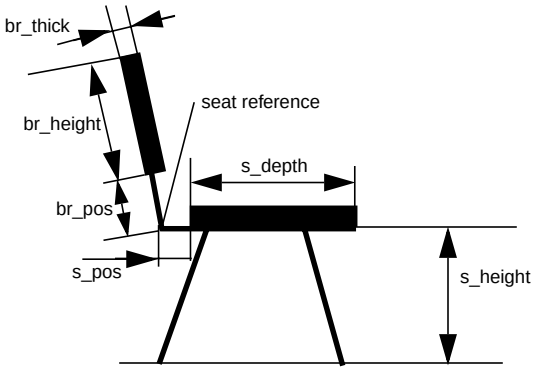

Figure 7. Schematic representation of chair parameters

The ergonomic critic will also make available the maximal allowed range for the distance between the front of the seat and the seat reference (17 in.). Based on this information the seat designer decides to acquire an expectation about the distance between the back end of the seat and the seat reference (*s_pos*).

The frame design agent's preconditions are such that it will decide the parameters *s_pos* and *br_pos* (the analogous parameter for the backrest) only after the backrest and seat design are completed. Therefore, at the point where it decides *s_depth*, the seat designer cannot verify whether the constraint will be violated. Hence an expectation would be useful.

Once the critique is posted, the design progresses through a conflict resolution process, implemented as a relaxation of the decision taken by one of the two agents: in this case the seat designer. However, this is a solution only for that particular design session, and the situation will *re-occur* every time the design context repeats itself.

After a preset number of violations of the constraint, the seat designer will initiate an expectation learning process for *s_pos*. The expectation will be used every time the agent has to make a decision for *s_depth*, and before posting it in the design system.

Alternatively, the seat designer might request an estimate for *s_pos* from the backrest designer. Since the backrest designer does not have the necessary information to carry out the computation (not necessarily only because of the seat designer), it will need in turn to substitute for the missing information through expectation learning.

This would transform the expectation learning of the seat designer for the purpose of evaluating the *consequences* of a decision into an expectation learning process of the backrest designer for the purpose of providing information for an *early* decision.

In the causal attribution phase the seat designer uses the design model and the design constraints to determine a set of candidate conditions that may impact the range of *s_pos.* The values of *s_pos* are discretized in ranges of 0.5 in, and each range will represent a class for the expectation target *s_pos.* Figure 8 illustrates the five candidate condition identified by the seat designer:

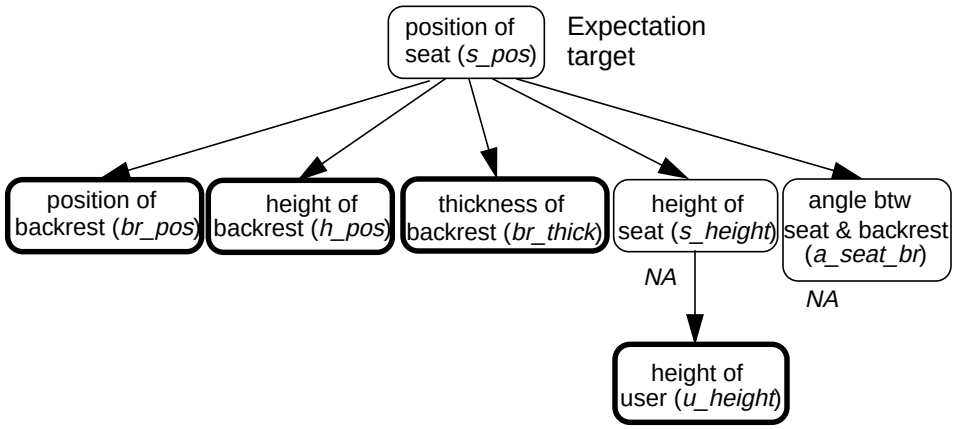

Figure 8. Candidate conditions for the expectation target identified through the causal attribution process (*NA* = not available)

The first three parameters are decided by the backrest agent, which has preceded the seat agent in the design process, and are therefore known. The last two parameters will be decided by the frame designer and are unavailable at this point.

The seat designer will attempt to further the causal attribution process for the two unavailable parameters. The only available information that it has to continue this process is a constraint that relates the height of the user *u_height* with *s_pos*+*s_depth*+*s_height*, and therefore the agent conjectures that the height of

the user may causally influence the height of the seat. At the end of the causal attribution process the four candidate condition that will be subject to covariational analysis are *br_pos*, *h_pos*, *br_thick*, and *u_height*.

Once a learning process is started LEAD is run through a set of similar design problems to the one which has generated the expectation learning. For each design session, the agent will acquire the values of the candidate conditions *at the point where it would need to use the expectation*, and the value of the expectation target *after the design session has completed*. Each data set represents a training instance that will be used by the machine learning component included in the wrapper.

The LEAD wrapper component selects two conditions as part of the expectation condition set, the thickness of the backrest (*br_thick*), and the height of the user (*u_height*), as the subset of conditions that yields the highest prediction accuracy (in this case 93%). It should be noted, however, that the set of conditions is not accepted as valid if the prediction accuracy does not satisfy a minimal threshold determined by the developer at system design time.

7.0 Discussion and Conclusions

Clearly the approach we have introduced here needs to be validated in several respects. For example, given that the frame designer will proceed after the seat designer, it is possible that the frame designer agent will change its decision about *s_pos* based on the new values for *s_depth* proposed by the seat designer. This situation can be compensated for in the second validation phase (semantic validation) when the seat designer would validate the expectation with training instances collected after the expectation was used.

What happens if the expectation is invalidated in a design process where the expectation is used? Besides being used as a negative training instance for learning, the design agent implementation can retract the facts that were generated based on that expectation, and resume the design process without using the expectation.

Expectations summarize behaviors that are not explicitly represented anywhere else in the system, or at least are not available to the agent. Given their empirical status, expectations need to be evaluated primarily from the point of view of the quality of the design. This requires additional test runs, and verification on problems that differ in requirements. Although the learning process associated with an expectation stops after the validation has been successful, the agent needs to maintain the tentative character of the acquired knowledge, and be able to further verify its validity and revise its status if required.

Expectations are a vital component of a large high-quality design system. As the need for them cannot be completely determined in advance, they must be learned when a need is detected. Causal attribution and covariational analysis combine to provide a powerful technique for the formation of expectations.

8.0 References

Berker, I., and D.C. Brown (1996). Conflict and negotiations in single function agent based design systems. *Concurrent Engineering: Research and Applications*. Special issue on Multi-Agent Systems in Concurrent Engineering, D.C. Brown, S. Lander, and C. Petrie (eds.) (1):17-3.

Brown, D.C., B. Dunskus, and D. Grecu (1994). Using Single Function Agents to Investigate Negotiations. *AAAI-94 Workshop on Models of Conflict Management in Cooperative Problem Solving*, Seattle, WA.

Cherniak, C. (1986). *Minimal Rationality.* Cambridge, MA: The MIT Press.

Cranz, G. (1998). *The Chair–Rethinking Culture, Body, and Design*. New York, NY: W.W. Norton & Company.

Dunskus, B.V., D.L. Grecu, D.C. Brown, and I. Berker (1995). Using Single Function Agents to Investigate Conflict. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. Special issue on Conflict Management in Design, I. Smith (ed.). 9:299-312.

Gennari, J.H., P. Langley, and D. Fisher. (1989). Models of incremental concept formation. *Artificial Intelligence* 40:11-61.

Giarratano, J.C., and G. Riley. 1998. *CLIPS Reference Manual*. PWS Publishing Co.

Grecu, D.L., and D.C. Brown (1998). Dimensions of Learning in Design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. Special issue on Machine Learning in Design, A.H.B. Duffy, D.C. Brown, and A.K. Goel (eds.) 12 (April):117-122.

Grecu, D.L., and D.C. Brown (1998). Evaluating the Impact of Distributed Learning in Real-World Design Problems. *Workshop on Machine Learning in Design (5th International Conference on Artificial Intelligence in Design)*, Lisbon, Portugal.

Hilton, D.J. (1990). Conversational Processes and Causal Explanation. *Psychological Bulletin* 107 (1):65-81.

Kohavi, R., and G.H. John (1998). Wrappers for Feature Subset Selection. *Artificial Intelligence* 97 (1-2):273-324.

Koslowski, B. (1996). *Theory and Evidence: The Development of Scientific Reasoning*. Edited by L. Gleitman, S. Carey, E. Newport and E. Spelke, *Learning, Development, and Conceptual Change*. Cambridge, MA and London, UK: The MIT Press.

Lander, S. (1998). Issues in Multiagent Design Systems. *IEEE Expert*:18-26.

Liu, H., and R. Setiono (1998). Incremental Feature Selection. *Applied Intelligence*.

Malone, J. (1998). The Intelligent Synthesis Environment, *http://ise.larc.nasa.gov/*

McCormick, E. (1964). *Human Factors Engineering*, New York, NY: McGraw-Hill.

Russell, S., and E. Wefald (1991). *Do the Right Thing–Studies in Limited Rationality.* Cambridge, MA: The MIT Press.

SGI (1996). *MLC++ Utilities*. Silicon Graphics.

Shultz, T.R., G.W. Fisher, C.C. Pratt, and S. Rulf (1986). Selection of Causal Rules. *Child Development* 57:143-152.